Term Work 1

1 a

```
#include<unistd.h>
#include<stdio.h>
#include<sys/types.h>
#include<sys/wait.h>
int main()
{
        int fd[2],n;
        char buffer[100];
        pid_t p;
        pipe(fd);
        p=fork();
        if(p>0)
        {
                printf("Parent having pid %d\n", getpid());
                printf("My child's pid is %d\n", p);
                printf("Passing value to child\n");
                write(fd[1], "hello\n", 6);
        }
        else
        {
                printf("Child having pid %d\n", getpid());
                printf("My parent's pid is %d\n", getppid());
                n=read(fd[0], buffer, 100);
                printf("Child received data\n");
                write(1,buffer,n);
        }
}
```

1 b message queue read

```
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
struct mesg_buffer{
long mesg_type;
char mesg_text[100];
}message;

int main()
{
key_t key;
int msgid;
key=ftok("progfile",65);
msgid=msgget(key,0666|IPC_CREAT);

msgrcv(msgid,&message, sizeof(message),1,0);
printf("Data Recived is : %s \n",message.mesg_text);
msgctl(msgid, IPC_RMID, NULL);
return 0;
}
```

1 b message queue write

```
#include<sys/ipc.h>
#include<sys/msg.h>
#include<stdio.h>
#include<stdlib.h>
#define MAX 10
struct mesg_buffer{
long mesg_type;
char mesg_text[100];
}message;

int main()
{
key_t key ;
int msgid;
key=ftok("progfile",65);
msgid=msgget(key,0666 | IPC_CREAT);
message.mesg_type=1;
printf("Write Data");
fgets(message.mesg_text,MAX, stdin);
msgsnd(msgid,&message, sizeof(message),0);
printf("Data send is : %s \n",message.mesg_text);
return 0;
}
```

Term Work 2

Tcp client c

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

#define PORT 4444
void main(){
  int clientSocket;
  struct sockaddr_in serverAddr;
  char buffer[1024];
  clientSocket = socket(PF_INET, SOCK_STREAM, 0);
  printf("[+]Client Socket Created Sucessfully.\n");
  memset(&serverAddr, '\0', sizeof(serverAddr));
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_port = htons(PORT);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  connect(clientSocket, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
  printf("[+]Connected to Server.\n");
  recv(clientSocket, buffer, 1024, 0);
  printf("[+]Data Recv: %s\n", buffer);
  printf("[+]Closing the connection.\n");

}
```

Tcp server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>
#define PORT 4444
void main(){
  int sockfd;
  struct sockaddr_in serverAddr;
  int newSocket;
  struct sockaddr_in newAddr
  socklen_t addr_size;
  char buffer[1024];
  sockfd = socket(AF_INET, SOCK_STREAM, 0);
  printf("[+]Server Socket Created Sucessfully.\n");
  memset(&serverAddr, '\0', sizeof(serverAddr));
  serverAddr.sin_family = AF_INET;
  serverAddr.sin_port = htons(PORT);
  serverAddr.sin_addr.s_addr = inet_addr("127.0.0.1");
  bind(sockfd, (struct sockaddr*)&serverAddr, sizeof(serverAddr));
  printf("[+]Bind to Port number %d.\n", 4455);
  listen(sockfd, 5);
  printf("[+]Listening...\n");
  newSocket = accept(sockfd, (struct sockaddr*)&newAddr, &addr_size);
  strcpy(buffer, "Hello");
  send(newSocket, buffer, strlen(buffer), 0);
  printf("[+]Closing the connection.\n");
}
```

Term Work 3

```
#include<stdio.h>
#define nul 1000
#define nodes 10
int no;
struct node
{
        int a[nodes][4];
}router[nodes];

void init(int r)
{
        int i;
        for(i=1;i<=no;i++)
        {
                router[r].a[i][1]=i;
                router[r].a[i][2]=999;
                router[r].a[i][3]=nul;
        }
        router[r].a[r][2]=0;
        router[r].a[r][3]=r;
}


void inp(int r)
{
        int i;
        printf("\nEnter dist from the node %d to other nodes",r);
        printf("\nPls enter 999 if there is no direct route\n",r);
        for(i=1;i<=no;i++)
        {
                if(i!=r)
                {
                        printf("\nEnter dist to the node %d:",i);
                        scanf("%d",&router[r].a[i][2]);
                        router[r].a[i][3]=i;
                }
        }
}
```

```c
void display(int r)

{

    int i,j;

    printf("\n\nThe routing table for node %d is as follows:",r);

    for(i=1;i<=no;i++)

    {

        if(router[r].a[i][2]>=999)

            printf("\n\t\t\t %d \t no link \t no hop",router[r].a[i][1]);

        else

            printf("\n\t\t\t %d \t %d \t\t
%d",router[r].a[i][1],router[r].a[i][2],router[r].a[i][3]);

    }

}
```

```c
void display(int r)

{

    int i,j;

    printf("\n\nThe routing table for node %d is as follows:",r);

    for(i=1;i<=no;i++)

    {

        if(router[r].a[i][2]>=999)

            printf("\n\t\t\t %d \t no link \t no hop",router[r].a[i][1]);

        else

            printf("\n\t\t\t %d \t %d \t\t
%d",router[r].a[i][1],router[r].a[i][2],router[r].a[i][3]);

    }

}
```

```
void dv_algo(int r)
{
        int i,j,z;
        for(i=1;i<=no;i++)
        {
                if(router[r].a[i][2]!=999 && router[r].a[i][2]!=0)
                {
                        for(j=1;j<=no;j++)
                        {
                                z=router[r].a[i][2]+router[i].a[j][2];
                                if(router[r].a[j][2]>z)
                                {
                                        router[r].a[j][2]=z;
                                        router[r].a[j][3]=i;
                                }
                        }
                }
        }
}

int main()
{
        int i,j,x,y;
        char choice='y';
        printf("Enter the number of nodes:");
        scanf("%d",&no);
        for(i=1;i<=no;i++)
        {
                init(i);
                inp(i);
        }

        printf("\nThe configuration of the nodes after initialization is as follows:");
        for(i=1;i<=no;i++)
        display(i);
for(i=1;i<=no;i++)
        dv_algo(i);

        printf("\nThe configuration of the nodes after computation of paths is as follows:");
        for(i=1;i<=no;i++)
        display(i);

        while(choice!='n')
        {
        printf("\nEnter the nodes btn which shortest path is to be found:\n");
        scanf("%d %d",&x,&y);
        printf("\nThe length of the shortest path is %d",router[x].a[y][2]);
        printf("\n\n continue ? (y/n):");
        scanf("%s",&choice);
        }

}
```

Term Work 4

Server

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char **argv){

  if (argc != 2) {
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }

  char *ip = "127.0.0.1";
  int port = atoi(argv[1]);
  int sockfd;
  struct sockaddr_in server_addr, client_addr;
  char buffer[1024];
  socklen_t addr_size;
  int n;
  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  if (sockfd < 0) {
    perror("[-]socket error");
    exit(1);
  }

  memset(&server_addr, '\0', sizeof(server_addr));
  server_addr.sin_family = AF_INET;
  server_addr.sin_port = htons(port);
  server_addr.sin_addr.s_addr = inet_addr(ip);

  n = bind(sockfd, (struct sockaddr*)&server_addr, sizeof(server_addr));
  if (n < 0){
    perror("[-]bind error");
    exit(1);
  }

  bzero(buffer, 1024);
  addr_size = sizeof(client_addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&client_addr, &addr_size);
  printf("[+]Data recv: %s\n", buffer);

  bzero(buffer, 1024);
  strcpy(buffer, "Welcome to the UDP Server.");
  sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&client_addr, sizeof(client_addr));
  printf("[+]Data send: %s\n", buffer);

  return 0;
}
```

Client

```
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <sys/socket.h>
#include <sys/types.h>
#include <netinet/in.h>
#include <arpa/inet.h>

int main(int argc, char **argv){

  if (argc != 2) {
    printf("Usage: %s <port>\n", argv[0]);
    exit(0);
  }

  char *ip = "127.0.0.1";
  int port = atoi(argv[1]);

  int sockfd;
  struct sockaddr_in addr;
  char buffer[1024];
  socklen_t addr_size;

  sockfd = socket(AF_INET, SOCK_DGRAM, 0);
  memset(&addr, '\0', sizeof(addr));
  addr.sin_family = AF_INET;
  addr.sin_port = htons(port);
  addr.sin_addr.s_addr = inet_addr(ip);

  bzero(buffer, 1024);
  strcpy(buffer, "Hello World!");
  sendto(sockfd, buffer, 1024, 0, (struct sockaddr*)&addr, sizeof(addr));
  printf("[+]Data send: %s\n", buffer);

  bzero(buffer, 1024);
  addr_size = sizeof(addr);
  recvfrom(sockfd, buffer, 1024, 0, (struct sockaddr*)&addr, &addr_size);
  printf("[+]Data recv: %s\n", buffer);

  return 0;
}
```

Term Work 5

Same code as that for term work 2

TERMWORK 6 STEPS (NS3)

Step 1 : Open **UBUNTU** and locate and open **ns-allinone-3.28** folder on **Desktop**.

Step 2 : Go to **ns-3.28** folder and open **examples->tutorial->first.cc**

Step 3 : In **first.cc** , include the following code.

```
                    #include “ns3/netanim-module.h”

                    AnimationInterface anim(“first, xml”);

                    AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream(“first.tr”));

                    pointToPoint.EnablePcapAll(“first”);
```

Step 4 : Copy **first.cc** and paste it in **ns-3.28->scratch** folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratch-simulator.cc and other files can be deleted.

Step 5 : Open terminal and change working directory to Desktop by **cd Desktop** and type following commands to go to location where scratch executable file is located i.e. scratch folder.

Step 6 : cd ns-allinone-3.28

Step 7 : cd ns-3.28

Step 8 : Run the **first.cc** by entering following command.

```
                    ./waf –run scratch/first
```

Step 9 : Once build is successful, return to **ns-allinone-3.28** folder with **cd ../** and enter into **netanim-3.108** with **cd netanim-3.108**

Step 10 : Now to see the animation, we have to open NetAnim software. So open by entering **./NetAnim** on terminal.

Step 11 : In NetAnim, open **first, xml** by clicking on **open XML trace file** icon.

Step 12 : Click on **run option/icon** to see the animation. To see the packet transfer, open **Packets Tab**

TERMWORK 7 STEPS (NS3)

Step 1 : Open **UBUNTU** and locate and open **ns-allinone-3.28** folder on **Desktop**.

Step 2 : Go to **ns-3.28** folder and open **examples->tutorial->second.cc**

Step 3 : In **second.cc** , include the following code.

```
                #include “ns3/netanim-module.h”

                AnimationInterface anim(“second, xml”);

                AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream(“second.tr”));

                pointToPoint.EnablePcapAll(“second”);
```

Step 4 : Copy **second.cc** and paste it in **ns-3.28->scratch** folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratch-simulator.cc and other files can be deleted.

Step 5 : Open terminal and change working directory to Desktop by **cd Desktop** and type following commands to go to location where scratch executable file is located i.e. scratch folder.

Step 6 : cd ns-allinone-3.28

Step 7 : cd ns-3.28

Step 8 : Run the **second.cc** by entering following command.

```
                ./waf –run scratch/second
```

Step 9 : Once build is successful, return to **ns-allinone-3.28** folder with **cd ../** and enter into **netanim-3.108** with **cd netanim-3.108**

Step 10 : Now to see the animation, we have to open NetAnim software. So open by entering **./NetAnim** on terminal.

Step 11 : In NetAnim, open **second, xml** by clicking on **open XML trace file** icon.

Step 12 : Click on **run option/icon** to see the animation. To see the packet transfer, open **Packets Tab**.

TERMWORK 8 STEPS (NS3)

Step 1 : Open **UBUNTU** and locate and open **ns-allinone-3.28** folder on **Desktop**.

Step 2 : Go to **ns-3.28** folder and open **examples->tutorial->third.cc**

Step 3 : In **third.cc** , include the following code.

```
                #include “ns3/netanim-module.h”

                AnimationInterface anim(“third, xml”);

                AsciiTraceHelper ascii;

pointToPoint.EnableAsciiAll(ascii.CreateFileStream(“third.tr”));

                pointToPoint.EnablePcapAll(“third”);
```

Step 4 : Copy **third.cc** and paste it in **ns-3.28->scratch** folder. Remember that scratch folder should contain only one .cc example file and it must contain scratch executable file named scratch-simulator.cc and other files can be deleted.

Step 5 : Open terminal and change working directory to Desktop by **cd Desktop** and type following commands to go to location where scratch executable file is located i.e. scratch folder.

Step 6 : cd ns-allinone-3.28

Step 7 : cd ns-3.28

Step 8 : Run the **third.cc** by entering following command.

```
                ./waf –run scratch/third
```

Step 9 : Once build is successful, return to **ns-allinone-3.28** folder with **cd ../** and enter into **netanim-3.108** with **cd netanim-3.108**

Step 10 : Now to see the animation, we have to open NetAnim software. So open by entering **./NetAnim** on terminal.

Step 11 : In NetAnim, open **third, xml** by clicking on **open XML trace file** icon.

Step 12 : Click on **run option/icon** to see the animation. To see the packet transfer, open **Packets Tab**.

COOJA SIMULATOR

Termwork – 9

Step 1 : Go to the Location contiki-ng/tools/cooja/ with commands

```
Cd contiki-ng
Cd tools
Cd cooja
```

Step 2 : Run the **cooja** simulator with

```
ant run
```

This allows cooja simulator to run and the build messages will be shown on the terminal.

The cooja simulator window opens up.

Step 3 : Create a **new simulation** by clicking file menu present in **Files** Tab.

Step 4 : Click on **Motes** tab, and create **Sky mote** as

Add motes -> Create new mote type -> Sky mote

In the window opened, give the file name, and for **Contiki process/Firmware** browse the file **ipv6-hooks.c**. Select the same.

Click on **compile** button and create the motes by clicking on **create** button.

Step 5 : In the motes window opened, enter the number of motes you want to create. (Here mote refers to the node in the network). Keep all other options as they are.

The motes are shown on the Networks section.

Step 6 : Configure the motes. i. e. set the motes as server and client.

To do this, right click on any mote, select **mote tools for Sky3**, and select **Serial Socket (CLIENT)** if you want to set that mote as client or select **Serial Socket(SERVER)** to make the mote as server.

Step 7 : **Serial Socket Server and Client** windows appear.

Make the **client port number** same as that of the server.

Start the Server by clicking **Start** button in **Serial Socket Server** window, start the client by following the same in Serial.

Connect the client and server by clicking **connect** button in the client window. It shows the **connected** message in green color.

Step 8 : Start the connection by clicking **start** button in **Simulation control** window.

We can check the output in **Mote Output** Window.

Termwork – 10

For this, the whole proess remains the same, only following steps change.

Step 4 : Upload two files for udp client and udp server.

Create one mote for client and upload udp client file for it. Which is present in,

contiki-ng ->examples->rpl-udp

Upload **udp-client.c** for client and configure this mote as client as given in **Step 6** above.

Create One more mote and upload **udp-server.c** file for it with above procedure and configure this mote as the server.

Step 5 : Create only one mote for client and one mote for Server.